Available online at www.sciencedirect.com

ScienceDirect

Journal of Applied Logic 4 (2006) 305–330

JOURNAL OF APPLIED LOGIC

www.elsevier.com/locate/jal

A general tableau method for propositional interval temporal logics: Theory and implementation ☆

V. Goranko [a,*], A. Montanari [b], P. Sala [b], G. Sciavicco [b]

[a] *Department of Mathematics, University of Johannesburg, South Africa*
[b] *Department of Mathematics and Computer Science, University of Udine, Italy*

Available online 9 August 2005

Abstract

In this paper, we focus our attention on tableau methods for propositional interval temporal logics. These logics provide a natural framework for representing and reasoning about temporal properties in several areas of computer science. However, while various tableau methods have been developed for linear and branching time point-based temporal logics, not much work has been done on tableau methods for interval-based ones. We develop a general tableau method for Venema's CDT logic interpreted over partial orders (BCDT$^+$ for short). It combines features of the classical tableau method for first-order logic with those of explicit tableau methods for modal logics with constraint label management, and it can be easily tailored to most propositional interval temporal logics proposed in the literature. We prove its soundness and completeness, and we show how it has been implemented.

Keywords: Interval temporal logics; Proof systems; Tableau methods

☆ This paper is an extended and revised version of [V. Goranko, A. Montanari, G. Sciavicco, A general tableau method for propositional interval temporal logics, in: Proc. of the International Conference TABLEAUX 2003, in: Lecture Notes in Artif. Intell., vol. 2796, Springer, Berlin, 2003, pp. 102–116. [12]].

* Corresponding author.
E-mail addresses: vfg@rau.ac.za (V. Goranko), montana@dimi.uniud.it (A. Montanari), sala@dimi.uniud.it (P. Sala), sciavicc@dimi.uniud.it (G. Sciavicco).

1570-8683/$ – see front matter
doi:10.1016/j.jal.2005.06.012

1. Introduction

In this paper, we focus our attention on tableau methods for propositional interval temporal logics. These logics provide a natural framework for representing and reasoning about temporal properties in several areas of computer science. However, while various tableau methods have been developed for linear and branching time point-based temporal logics, e.g., [5,9,18,29,33], not much work has been done on tableau methods for interval-based temporal logics. One reason for this disparity is that operators of interval temporal logics are in many respects more difficult to deal with. As an example, there exist straightforward inductive definitions of the main operators of point-based temporal logics, such as the *future* and the *until* operators, while inductive definitions of basic interval modalities turn out to be much more complex (consider, for instance, the one for the *chop* operator given in [4]).

Various propositional and first-order interval temporal logics have been proposed in the literature (see [13] for an up-to-date survey that analyzes the main contributions in the field). Propositional interval temporal logics include Halpern and Shoham's Modal Logic of Time Intervals (HS) [17], Venema's CDT logic [32], Moszkowski's Propositional Interval Temporal Logic (PITL) [22], and Goranko, Montanari, and Sciavicco's family of Propositional Neighborhood Logics ($\mathcal{PNL}$) [11], while the most interesting first-order versions are Moszkowski's Interval Temporal Logic (ITL) [22] and Zhou and Hansen's Neighborhood Logic (NL) [36]. Two different semantics have been given to interval logics, namely, a *non-strict* one, which includes intervals with coincident endpoints (point-intervals), and a *strict* one, which excludes them. We restrict our attention to the propositional setting, and we assume the non-strict semantics as the default (it is the most common and general).

In this paper, we develop a sound and complete general tableau method for Venema's CDT logic interpreted over partial orders, called (Non-Strict) Branching CDT (BCDT$^+$ for short). BCDT$^+$ features the same operators as CDT; however, since it is interpreted over partially ordered domains with linear intervals, it is expressive enough to include as subsystems or specializations all the above-mentioned propositional interval logics. While most existing tableau methods for modal and temporal logics are terminating methods for *decidable* logics, and thus they yield decision procedures, the proposed tableau method for BCDT$^+$ only provides a semi-decision procedure for unsatisfiability. In this respect, even though it shares some basic features with explicit tableaux for modal logics with constraint label management, it comes closer to the classical, possibly non-terminating tableau method for first-order logic [8]. Furthermore, it presents some similarities with the explicit tableau method developed for the *guarded fragment* of first-order logic [14]. Finally, it can be easily adapted to variations and subsystems of BCDT$^+$, thus providing a general tableau method for propositional interval logics.

The rest of the paper is organized as follows. In Section 2, we introduce the syntax and semantics of BCDT$^+$, and we compare its expressive power with that of the main propositional interval logics. In Section 3, we provide a survey of existing tableau methods for propositional temporal logics. In Section 4, we present our tableau method, and we prove its soundness and completeness. In Section 5, we describe the basic features of an

efficient implementation of the method in an imperative language. Conclusions provide an assessment of the work and outline future research directions.

2. CDT over partial orders (BCDT$^+$)

In this section, we give syntax and semantics of BCDT$^+$ and discuss its expressive power. To this end, we introduce some preliminary notions. Let $\mathbb{D} = \langle \mathrm{D}, < \rangle$ be a strict partial order. A (non-strict) *interval* on $\mathbb{D}$ is an ordered pair $[d_0, d_1]$ such that $d_0, d_1 \in \mathrm{D}$, and $d_0 \leqslant d_1$. When $d_0 < d_1$ we say that the interval is *proper* or *strict*; when $d_0 = d_1$ it is a *point-interval*. The set of all non-strict intervals on $\mathbb{D}$ will be denoted by $\mathbb{I}(\mathbb{D})^+$, while the set of all strict intervals will be denoted by $\mathbb{I}(\mathbb{D})^-$; by $\mathbb{I}(\mathbb{D})$ we will denote either of these. As in [17], we assume intervals to be *linear*, that is, for every interval $[d_0, d_1]$ and every pair of points d, d' belonging to it, namely, $d_0 \leqslant d \leqslant d_1$ and $d_0 \leqslant d' \leqslant d_1$, $d < d'$ or $d' < d$ or $d = d'$. Such an assumption keeps the temporal setting still very general, while making it fitting our intuition about the nature of time. In Fig. 1 we give an example of a non-linear interval structure with the linear interval property (left) and an example of a non-linear interval structure that does not satisfy it (right). A pair $\langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle$ is called an *interval structure*. We can constrain an interval structure to be *linear*, *branching*, *discrete*, *dense*, *unbounded above* and/or *below*, *Dedekind complete*, and so on, by imposing suitable conditions on it. An element $d \in \mathrm{D}$ such that there are no elements $d' \in \mathrm{D}$ with $d < d'$ (resp., $d' < d$) is called *minimal* (resp., *maximal*) element.

BCDT$^+$ features the same operators as CDT, but it is interpreted over partially ordered domains with linear intervals. Its language consists of a set of propositional variables $\mathcal{AP}$, the logical connectives $\neg$ and $\wedge$, the modalities C, D, and T, and the modal constant π. The other logical connectives, as well as the logical constants $\top$ and $\bot$, can be defined in the usual way. BCDT$^+$ well-formed *formulas*, denoted by $\phi, \psi, \ldots$, are recursively defined as follows (where $p \in \mathcal{AP}$):

$$\phi = \pi \mid p \mid \neg\phi \mid \phi \wedge \psi \mid \phi C \psi \mid \phi D \psi \mid \phi T \psi.$$

The semantics of BCDT$^+$ is given in terms of *non-strict models* of the type $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(D)^+, \mathcal{V} \rangle$, based on non-strict interval structures, where $\mathcal{V}$ is a *valuation function* for propositional variables. The valuation function is a mapping $\mathcal{V}: \mathbb{I}(D)^+ \mapsto 2^{\mathcal{AP}}$, where $\mathbb{I}(D)^+$ is the set of all intervals in $\mathbb{D}$, such that, for any $p \in \mathcal{AP}$, p holds over $[d_0, d_1]$ if and only if $p \in \mathcal{V}([d_0, d_1])$. *Truth* over an interval $[d_0, d_1]$ in a model $\mathbf{M}^+$ is defined by induction on the structure of formulas:

- $\mathbf{M}^+, [d_0, d_1] \Vdash \pi$ iff $d_0 = d_1$;

Fig. 1. A non-linear interval structure with/without the linear interval property.

- $\mathbf{M}^+, [d_0, d_1] \Vdash p$ iff $p \in \mathcal{V}([d_0, d_1])$, for all $p \in \mathcal{AP}$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \neg\psi$ iff it is not the case that $\mathbf{M}^+, [d_0, d_1] \Vdash \psi$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \phi \wedge \psi$ iff $\mathbf{M}^+, [d_0, d_1] \Vdash \phi$ and $\mathbf{M}^+, [d_0, d_1] \Vdash \psi$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \phi C \psi$ iff there exists $d_2 \in \mathrm{D}$ such that (i) $d_0 \leqslant d_2 \leqslant d_1$, and (ii) $\mathbf{M}^+, [d_0, d_2] \Vdash \phi$ and $\mathbf{M}^+, [d_2, d_1] \Vdash \psi$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \phi D \psi$ iff there exists $d_2 \in \mathrm{D}$ such that (i) $d_2 \leqslant d_0$, and (ii) $\mathbf{M}^+, [d_2, d_0] \Vdash \phi$ and $\mathbf{M}^+, [d_2, d_1] \Vdash \psi$;
- $\mathbf{M}^+, [d_0, d_1] \Vdash \phi T \psi$ iff there exists $d_2 \in \mathrm{D}$ such that (i) $d_1 \leqslant d_2$, and (ii) $\mathbf{M}^+, [d_1, d_2] \Vdash \phi$ and $\mathbf{M}^+, [d_0, d_2] \Vdash \psi$.

Satisfiability and *validity* of BCDT^+ formulas are defined in the usual way.

Let us compare the expressive power of BCDT^+ with that of the main propositional interval logics proposed in the literature. We say that a logic $\mathbf{L}_1$ is *at least as expressive* as a logic $\mathbf{L}_2$ if for every $\mathbf{L}_2$ formula there exists an equivalent $\mathbf{L}_1$ formula, and that $\mathbf{L}_1$ is (strictly) *more expressive* than $\mathbf{L}_2$ if and only if $\mathbf{L}_1$ is at least as expressive as $\mathbf{L}_2$, but not vice versa.

We preliminarily summarize the main characteristics of the considered interval temporal logics. HS features four basic operators: $\langle B \rangle$ (*begins*) and $\langle E \rangle$ (*ends*), and their transposes $\langle \overline{B} \rangle$ (*begun by*) and $\langle \overline{E} \rangle$ (*ended by*). Given a formula ϕ and an interval $[d_0, d_1]$, $\langle B \rangle \phi$ holds over $[d_0, d_1]$ if ϕ holds over $[d_0, d_2]$, for some $d_0 \leqslant d_2 < d_1$, and $\langle E \rangle \phi$ holds over $[d_0, d_1]$ if ϕ holds over $[d_2, d_1]$, for some $d_0 < d_2 \leqslant d_1$. It is possible to show that HS captures all Allen's relations [2]. In particular, it allows one to define the *strict after* operator $\langle A \rangle$ (and its transpose $\langle \overline{A} \rangle$) such that $\langle A \rangle \phi$ holds over $[d_0, d_1]$ if ϕ holds over $[d_1, d_2]$ for some $d_2 > d_1$, the *non-strict after* operator $\Diamond_r$ (and its transpose $\Diamond_l$) such that $\Diamond_r \phi$ holds over $[d_0, d_1]$ if ϕ holds over $[d_1, d_2]$ for some $d_2 \geqslant d_1$, and the *sub-interval* operator $\langle D \rangle$ such that $\langle D \rangle \phi$ holds over a given interval $[d_0, d_1]$ if ϕ holds over a proper sub-interval of $[d_0, d_1]$. In [17], Halpern and Shoham have shown the undecidability of HS over various classes of linear orders by a suitable encoding of the halting problem.

CDT has three binary operators C (*chop*), D, and T, which correspond to the ternary interval relations occurring when an extra point is added in one of the three possible distinct positions with respect to the two endpoints of the current interval (*between, before*, and *after*), plus a modal constant π which holds over a given interval if and only if it is a point-interval. Since HS can be embedded into CDT, undecidability results for the latter follow from those for the former.

PITL features the two modalities $\bigcirc$ (*next*) and C (the specialization of the *chop* operator for discrete structures). Intervals are defined as finite or infinite sequences of states. Given two formulas ϕ, ψ and a (finite) interval $d_0, \ldots, d_n$, $\bigcirc\phi$ holds over $d_0, \ldots, d_n$ if and only if ϕ holds over $d_1, \ldots, d_n$, while $\phi C \psi$ holds over $d_0, \ldots, d_n$ if and only if there exists i, with $0 \leqslant i \leqslant n$, such that ϕ holds over $d_0, \ldots, d_i$ and ψ holds over $d_i, \ldots, d_n$. PITL has been proved to be undecidable by a reduction from the problem of testing the emptiness of the intersection of two grammars in Greibach form [22]. A decidable fragment of PITL extended with quantification over propositional variables (QPITL) has been obtained by imposing a suitable *locality* constraint which states that each propositional variable is true over an interval if and only if it is true at its first state [22]. By exploiting such a con-

straint, decidability of Local QPITL can be easily proved by embedding it into quantified propositional Linear Temporal Logic.

Finally, propositional neighborhood logics in $\mathcal{PNL}$ have two modalities for right and left interval neighborhoods, namely, $\langle A\rangle$ and $\langle \overline{A}\rangle$ in the strict semantics ($\mathcal{PNL}^-$ logics), and $\Diamond_r$ and $\Diamond_l$ in the non-strict semantics ($\mathcal{PNL}^+$ logics). While the undecidability of the first-order Neighborhood Logic NL can be easily proved by embedding HS in it, the decidability problem for its propositional fragments is still open.

We first note that both CDT and non-strict Propositional Neighborhood Logics ($\mathcal{PNL}^+$) are interpreted over linear structures, and that the operators of $\mathcal{PNL}^+$ logics can be expressed in CDT by means of the formulas $\Diamond_r\phi := \phi T\top$ and $\Diamond_l\phi := \phi D\top$. Moreover, it is well known that CDT does not semantically include HS in its full generality, since the latter allows the interval structure to be branching, while the former does not. On the other hand, HS is not more expressive than CDT, because it cannot express the *chop* operator [21].

BCDT$^+$ generalizes Venema's CDT (and thus all logics in $\mathcal{PNL}^+$) by allowing the interval structure to be non-linear, for as long as all intervals in it are linear (as in HS). Furthermore, it is strictly more expressive than HS and PITL. HS operators can be defined in BCDT$^+$ as follows: $\langle B\rangle\phi := \phi C\neg\pi$, $\langle \overline{B}\rangle\phi := \neg\pi T\phi$, $\langle E\rangle\phi := \neg\pi C\phi$, and $\langle \overline{E}\rangle\phi := \neg\pi D\phi$. Besides, the strict neighborhood operators $\langle A\rangle$ and $\langle \overline{A}\rangle$ can be defined in BCDT$^+$ by using π as follows: $\langle A\rangle\phi := (\phi \wedge \neg\pi)T\top$, and $\langle \overline{A}\rangle\phi := (\phi \wedge \neg\pi)D\top$. By exploiting such derived operators, all conditions on the interval structure mentioned in the preliminaries can be easily expressed in BCDT$^+$. In particular, linearity can be expressed in BCDT$^+$ by means of the following formula:

$$lin \triangleq \big(\langle A\rangle p \to [A]\big(p \vee \langle B\rangle p \vee \langle \overline{B}\rangle p\big)\big) \wedge \big(\langle \overline{A}\rangle p \to [\overline{A}]\big(p \vee \langle E\rangle p \vee \langle \overline{E}\rangle p\big)\big),$$

while discreteness of linear interval structures can be imposed by means of the formula:

$$disc \triangleq \pi \vee l1 \vee \big(\langle B\rangle l1 \wedge \langle E\rangle l1\big),$$

where $l1$ stands for $\langle B\rangle\top \wedge [B][B]\bot$.

As for the PITL operators, C is an operator of BCDT$^+$, while $\bigcirc$ can be defined over (linear) discrete structures as follows: $\bigcirc\phi := l1C\phi$. On the other hand, BCDT$^+$ is strictly more expressive than PITL, since the latter is not able to access any interval which is not a sub-interval of the current interval.

The undecidability of BCDT$^+$ with respect to a number of interval structures immediately follows from results in [17], while finding meaningful decidable fragments of BCDT$^+$ is an interesting open problem.

3. Tableau methods for temporal logics

In this section, we survey existing tableau methods for propositional point-based and interval-based temporal logics over linear and branching time. According to [8], tableau methods for (modal and) temporal logics can be classified as *explicit* or *implicit*. Explicit methods keep track of the accessibility relation by means of some sort of external device. One possibility is to maintain an auxiliary graph of named nodes $\mathbf{n}_i, \mathbf{n}_j, \ldots$, where each

node contains a subformula, or a set of subformulas, of the formula to be checked, and the existence of an edge from $\mathbf{n}_i$ to $\mathbf{n}_j$ means that $\mathbf{n}_j$ is accessible from $\mathbf{n}_i$. Another possibility is to include structured labels into nodes to constrain the formula, or the set of formulas, associated with each node to hold only at the domain element(s) identified by the label. The resulting labeled tableau systems capture the accessibility relation by means of labeled formulas, and they provide suitable notions of closed branches and tableaux. In implicit methods [10,27], the accessibility relation is built-in into the structure of the tableau. As an example, in the case of linear and branching time point-based temporal logics the tableau represents a model of the satisfiable formulas (a timeline or a tree, respectively). The non-standard finite model property can then be exploited to show that the resulting tableau methods are actually decision procedures (they do not lead to infinite computations). In [18], implicit methods are further partitioned into *declarative* and *incremental* ones. Methods in the former class first generate all possible sets of subformulas of a given formula, and then they eliminate some (possibly all) of them, while those in the latter generate only 'meaningful' sets of subformulas.

3.1. Point-based linear and branching temporal logics

The problem of devising tableau systems for propositional Linear Temporal Logic (LTL), as well as for some extensions and fragments of it, has been extensively investigated in the literature. An exponential time declarative method to check LTL formulas has been developed by Wolper [33] and later extended by Lichtenstein and Pnueli to Past LTL (PLTL) [26], while an incremental method for PLTL has been proposed by Kesten et al. [18]. A labeled tableau system for the LTL-fragment LTL[F] has been proposed by Schmitt and Goubault-Larrecq [29] (an attempt to extend it to full LTL is reported in [30]). Finally, a tableau method for PLTL over bounded models has been developed by Cerrito and Cialdea-Mayer [5] (in [6], Cerrito et al. generalize the method to first-order PLTL). The satisfiability problem for LTL and PLTL is PSPACE-complete [31], while that for PLTL over bounded models of polynomial length and LTL[F] is NP-complete [5,31].

Wolper's tableau method is a natural extension to the one for propositional logic. The key idea is to take advantage of the so-called fix-point definition of temporal operators that allows one to split every temporal formula into a (possibly empty) part related to the current state and a part related to the next (resp. previous) state. As an example, the formula $\phi U \psi$ is analyzed as follows: either ψ holds now, or ϕ holds now and $\phi U \psi$ holds at the next state. Since only a finite set of distinct scenarios can be generated in this way, it is possible to devise a mechanism to control the repeated appearances of formulas, and to identify periodic situations in a finite time. The algorithm for checking the (un)satisfiability of a PLTL-formula ϕ behaves as follows: first (*construction*), it builds the tableau for ϕ; then (*elimination*), it removes unsuitable maximal strongly connected components (maximal strongly connected components which are not reachable from an initial node including ϕ or are not self fulfilling and have not outgoing edges [33]). It turns out that the formula ϕ is satisfiable if and only if the elimination phase does not end with an empty tableau. In [18] Kesten et al. provide an efficient incremental variant of Wolper's declarative procedure, which extends to PLTL the incremental method for LTL originally developed by Pnueli and Sherman [25]. Its basic ingredients are the same as Wolper's. However, instead of

preliminarily generating the set of all nodes of the tableau and thus immediately paying the worst case exponential complexity price, it builds the tableau incrementally by introducing only those nodes which are reachable from an initial nodes including the formula to be checked. Even though in the worst case this procedure takes exponential time, one can expect that in many cases a much smaller number of nodes is explored.

Unlike the above-described implicit methods, the labeled tableau systems for LTL[F] and for PLTL over bounded models, respectively developed by Schmitt and Goubault-Larrecq [29] and by Cerrito and Cialdea-Mayer [5], employ a mechanism for labelling formulas with temporal constraints somewhat similar to ours.

The distinctive feature of Schmitt and Goubault-Larrecq's tableau system is that its termination can be established *locally*: it terminates when no further expansion rule can be applied, which is guaranteed to happen. The basic notions are those of signed clause, borrowed from [16], and temporal constraint. A *signed clause* is either a pair $[d_i, d_j]\Theta$ or a pair $|\infty|\Theta$, where Θ is a (multi)set of formulas (the *clause*) and $[d_i, d_j]$ is a *time interval*, where d_j can possibly be ∞. A signed clause $[d_i, d_j]\Theta$ evaluates to true in a given structure if for every $d \in [d_i, d_j]$, there exists $\psi \in \Theta$ such that ψ holds at d (disjunctive interpretation), while $|\infty|\Theta$ is true if for infinitely many time points d, every $\psi \in \Theta$ holds at d (conjunctive interpretation). A temporal constraint is an expression of the form $d_i \leqslant d_j$. A tableau T is a set of branches, where a *branch* is a pair (B, K) consisting of a set of signed clauses B and a set of temporal constraints K. The construction of the tableau is accomplished by applying two kinds of steps: *expansion* and *closing* steps. An expansion step is performed by choosing a branch (B, K) and an unused signed clause $\Theta \in B$, and by applying a matching logical tableau *rule*. As a result of this application, the premise clause is marked as used and the branch (B, K) is extended by adding the signed clauses and the constraints, possibly involving new states, in the conclusions of the rule to B and K, respectively. Whenever the conclusions include various alternatives, the branch is split accordingly. A signed clause such that Θ includes only literals is called *atomic*. A branch whose unused signed clauses are all atomic is called atomic, and it cannot be further expanded. An atomic branch (B, K) is *satisfiable* if there exists an interpretation that satisfies all signed clauses in B and the set of constraints in K; otherwise, it is unsatisfiable. A closing step is applied to an atomic branch, and it marks the branch as *closed* if it is unsatisfiable. Termination is proved by introducing a suitable complexity measure and by showing that, for every rule, every conclusion is smaller than the premise with respect to such a measure. Critical formulas, such as $GF\phi$, which potentially lead to infinite computations, are dealt with by using the symbol ∞ in a suitable way.

PLTL interpreted over bounded models, that is, finite sequences $d_0, d_1, \ldots, d_k$ of states where k is known in advance, has been introduced to address planning problems in AI. A tableau method for it has been developed by Cerrito and Cialdea-Mayer [5]. The boundaries of the model are encoded by means of the special constant symbols *start* and *finish*. A state d_i is encoded by means of an expression of the form $c + n$, where c is a constant and n is a natural number. A tableau is a set of branches to which *expansion* and *conflict resolution* rules are applied. Tableau nodes are either temporal constraints or labeled formulas. *Temporal constraints* are expressions of the form $d_i \leqslant d_j$, where d_i, d_j are states. An *initial constraint finish* $\leqslant$ *start* $+ k$, where k is the length of the model, is associated with every tableau. A *labeled formula* is a pair $([d_i, d_j], \psi)$, where $[d_i, d_j]$ is an interval

and ψ is a PLTL-formula, which states that ψ is true at every state between d_i and d_j (conjunctive interpretation). Expansion rules are applied to pairs of nodes of the forms $([d_i, d_j], \psi)$ and $d_i \leqslant d_j$, and they cause the expansion, and possibly the splitting, of the branch. Conflict resolution rules force the two intervals over which contradictory literals hold (if any) to be disjoint. The closure of a branch B is established by checking the set K of constraints associated with it (which includes the initial constraint): B is closed if and only if K is unsatisfiable. Termination, soundness, and completeness of the method are proved by exploiting a suitable notion of *canonical tableau*.

The main differences between these tableau methods and ours are: (i) they are specifically designed to deal with natural/integer time structures (i.e., linear and discrete), while ours makes no assumptions; (ii) intervals only play a secondary role in them (e.g., in Cerrito and Cialdea-Mayer's system a formula is true on an interval if and only if it is true at every point in it), while in our system intervals are primary semantic objects on which the truth definitions are entirely based; (iii) the closedness of the tableau is defined in terms of unsatisfiability of the associated set of temporal constraints, while in our system it is entirely syntactic.

We conclude this section by considering the satisfiability problem for CTL, which is known to be EXPTIME-complete. An implicit tableau method to check the satisfiability of CTL formulas, that generalizes Wolper's method for LTL, has been proposed by Emerson and Halpern in [9]. The algorithm for checking the (un)satisfiability of a CTL-formula ϕ basically behaves as Wolper's one: first, it builds the tableau for ϕ; then, it removes unsuitable maximal strongly connected components. The elimination phase encompasses both a local pruning process, that removes local inconsistencies, and another pruning process, that removes nodes including requests which are not fulfilled in the current tableau. As in the case of LTL, the formula ϕ is satisfiable if and only if the elimination phase does not end with an empty tableau.

3.2. Interval-based linear temporal logics and duration calculi

To the best of our knowledge, there exist very few tableau methods for interval temporal logics (and duration calculi) in the literature. A tableau-based decision procedure for an extension of Local PITL interpreted over finite state sequences (LPITL$_{proj}$), which pairs the operators $\bigcirc$ and C with a projection operator $proj$, has been proposed by Bowman and Thompson [4]. Such a procedure refines a previous tableau system for quantified LPITL$_{proj}$ developed by Kono [19]. It rests on a normal form for LPITL$_{proj}$ formulas that allows one to exploit a classical tableau method, devoid of any mechanism for constraint label management. In [7], Chetcuti-Serandio and Fari nas del Cerro isolate a fragment of Propositional Duration Calculus (PDC$_{pos}$), which only includes PDC formulas that satisfy suitable syntactic restrictions. PDC$_{pos}$ is expressive enough to capture Allen's relations [2] and decidable. The tableau construction for PDC$_{pos}$ combines the application of the rules of classical tableaux with that of a suitable constraint resolution algorithm and it essentially depends on the assumption of *bounded* variability of state expressions (they may have only a finite number of discontinuities on a bounded interval, thus being Riemann-integrable on all bounded intervals).

LPITL$_{proj}$ extends LPITL with the binary operator *proj* which yields general repetitive behavior. For any given pair of formulas ϕ and ψ, $\phi\, proj\, \psi$ holds over an interval if such an interval can be partitioned into a series of sub-intervals each of which satisfies ϕ, while ψ (called the *projected formula*) holds over the new interval collecting the endpoints of these sub-intervals. LPITL$_{proj}$ formulas are interpreted over finite state sequences $d_0, d_1, \ldots, d_k$. The valuation function $\mathcal{V}$ maps each interval $[d_i, d_j]$ into the set of propositional variables that hold over it. The locality constraint imposes that, for any propositional variable p and interval $[d_i, d_j]$, $p \in \mathcal{V}([d_i, d_j])$ if and only if $p \in \mathcal{V}([d_i, d_i])$. The problem of satisfiability checking for LPITL$_{proj}$ is non-elementary [13]. The core of Bowman and Thompson's tableau method is the definition of suitable normal forms for all operators of the logic, which reflect the locality constraint and provide the operators with uniform inductive definitions. Taking advantage of them, Bowman and Thompson develop an implicit tableau-based decision procedure for satisfiability checking in the style of Wolper's one [33]. The normal form for LPITL$_{proj}$ formulas has the following general format:

$$(\pi \wedge \phi_e) \vee \bigvee_i (\phi_i \wedge \bigcirc \phi_i'),$$

where π stands for the formula $\bigcirc \bot$ characterizing point-intervals, ϕ_e and ϕ_i are point formulas, and ϕ_i' is an arbitrary LPITL$_{proj}$ formula. The first disjunct states when a formula is satisfied over a point interval, while the second one states the possible ways in which a formula can be satisfied over a strict interval, namely, a point formula must hold at the initial point and then an arbitrary formula must hold over the remainder of the interval. This normal form embodies a recipe for evaluating LPITL$_{proj}$ formulas: the first disjunct is the base case, while the second disjunct is the inductive step. Bowman and Thompson show that any LPITL$_{proj}$ formula can be equivalently transformed into this normal form. As in the case of implicit methods for point-based temporal logics, the tableau construction splits the requirements imposed by any temporal formula into requirements about the present (the first state of an interval) and requirements about the remainder of the interval, and it generates a directed graph $G = (N, E)$, where each node corresponds to a state of the sequence and is labeled by a set of formulas. The construction of the graph G for a formula ϕ starts with the *initial* node $\mathbf{n}_0$ labeled with the set $\{\phi, \top C \pi\}$. The expansion rules for the Boolean connectives are the standard ones; formulas of the forms $\psi C \theta$ and $\psi\, proj\, \theta$, as well as $\neg(\psi C \theta)$ and $\neg(\psi\, proj\, \theta)$, are expanded by exploiting the normal forms of their subformulas; finally, as in Wolper's tableau method, formulas of the form $\bigcirc \psi$ are expanded into a new node, corresponding to a new state, labeled with ψ. Once the construction of the graph G has been completed, the procedure looks for unsatisfiable nodes in G and marks them. Unsatisfiable nodes are (i) nodes which contain a formula and its negation, (ii) nodes which contain both a formula $\bigcirc \psi$ and the formula π, and (iii) nodes whose successors are unsatisfiable. The formula ϕ is satisfiable if and only if the initial node is not marked. The proofs of termination, soundness, and completeness are similar to those for PLTL in their structure, but they are much more involved.

Chetcuti-Serandio and Fari nas del Cerro's tableau method operates on a decidable fragment of (propositional) Duration Calculus (DC). DC is a first-order interval temporal logic, interpreted over the set of reals, which is based on ITL [35,36]. The first-order language for DC extends the propositional one essentially the same way as in classical logic, but

accounting for the fact that the first-order domain may change over time. Among the constants, there is a specific and important one, that is, the constant l, whose interpretation can vary over time, denoting the *length of the current interval*. It is combined with the structure of the additive group of reals as part of the temporal domain, which allows, for instance, to compute the length of concatenated intervals. A specific additional feature of the syntax of DC is the special category of terms called *state expressions* which are used to represent the duration for which a system stays in a particular state. Chetcuti-Serandio and Fari nas del Cerro provide a tableau method for PDC_{pos} that presents many similarities with the one of Cerrito and Cialdea-Mayer. Tableau nodes are conjunctions of labeled formulas, labeled state expressions, and constraints (not all these components are necessarily included in any node). Labeled formulas (resp., state expressions) are pairs $(\phi, [d_i, d_j])$ (resp., $(\sigma, [d_i, d_j])$), where ϕ (resp., σ) is a formula (resp., state expression) and $[d_i, d_j]$ is an interval. Constraints can be either *qualitative*, e.g., $d_i \leqslant d_j$, and *quantitative*, e.g., $d_j - d_i = k$ or $d_j - d_i > k$, where k is a constant. The construction of the tableau is fairly standard. It starts with an initial node including the pair $(\phi, [d_0, d_1])$, where ϕ is the formula to be checked and $[d_0, d_1]$ is a generic interval, and it proceeds by applying suitable expansion rules to labeled formulas or labeled state expressions in the leaf node of the considered branch. Closing rules detect contradictory formulas associated with the same interval or inconsistent sets of constraints in a leaf node. The proof of termination basically exploits a lemma showing that each expansion rule can be applied finitely often to any branch, while the soundness and completeness proof takes advantage of a lemma showing that expansion rules preserve (a suitable notion of) satisfiability. Complexity issues are not addressed.

3.3. Miscellany

We conclude the section by mentioning some additional tableau systems that present interesting connections to ours, such as the tableau methods for temporal logics of knowledge and belief, the free-variable tableau methods for modal logics, the tableau methods for first-order temporal logics, and the generic tableau provers, such as Paulson's [24]. Tableau methods for the propositional temporal Logics of knowledge and belief KL_n and BL_n are described in [23,34]. They are implicit methods, like those for PLTL, that introduce a specialized accessibility relation and specific rules for agent management. Free-variable semantic tableaux are a well-established technique for first-order theorem proving. In [3], Beckert and Goré show that they can be exploited to deal with propositional modal logics, providing a compact, efficient, and easily implementable technique. Tableau methods for decidable (monodic) fragments of first-order temporal logic over the natural numbers have been developed by Kontchakov et al. [20]. The decision procedure is obtained by separating the temporal and the first-order components of the formula to be checked and by dealing with the former using tableau methods for LTL, and with the latter using existing procedures for first-order decidable fragments. Finally, Abate and Goré [1] propose a tableau workbench that allows one to easily derive implicit tableau methods for various systems of modal logics by specifying the appropriate expansion rules. Furthermore, it allows one to express side conditions for rule firing and to maintain the history of the applied expansion steps.

4. A tableau method for BCDT$^+$

In this section we devise a tableau method for BCDT$^+$. The method can be adapted to its strict version BCDT$^-$, and can be accordingly restricted to CDT, HS, PITL, and some $\mathcal{PNL}$ logics. We first introduce some basic terminology. A *finite tree* is a finite directed connected graph in which every node, apart from one (the *root*), has exactly one incoming edge. A *successor* of a node $\mathbf{n}$ is a node $\mathbf{n}'$ such that there is an edge from $\mathbf{n}$ to $\mathbf{n}'$. A *leaf* is a node with no successors; a *path* is a sequence of nodes $\mathbf{n}_0, \ldots, \mathbf{n}_k$ such that, for all $i = 0, \ldots, k-1$, $\mathbf{n}_{i+1}$ is a successor of $\mathbf{n}_i$; a *branch* is a path from the root to a leaf. The *height* of a node $\mathbf{n}$ is the maximum length (number of edges) of a path from $\mathbf{n}$ to a leaf. If $\mathbf{n}, \mathbf{n}'$ belong to the same branch and the height of $\mathbf{n}$ is less than (resp. less than or equal to) the height of $\mathbf{n}'$, we write $\mathbf{n} \prec \mathbf{n}'$ (resp. $\mathbf{n} \preceq \mathbf{n}'$).

Definition 1. If $\mathbb{C} = \langle C, < \rangle$ is a finite partial order, a *labeled formula*, with label in $\mathbb{C}$, is a pair $(\phi, [c_i, c_j])$, where $\phi \in$ BCDT$^+$ and $[c_i, c_j] \in \mathbb{I}(C)^+$. For a node $\mathbf{n}$ in a tree $\mathcal{T}$, the *decoration* $\nu(\mathbf{n})$ is a triple $((\phi, [c_i, c_j]), \mathbb{C}, u_{\mathbf{n}})$, where $(\phi, [c_i, c_j])$ is a labeled formula, with label in $\mathbb{C}$, and $u_{\mathbf{n}}$ is a *local flag function* which associates the values 0 or 1 with every branch B in $\mathcal{T}$ containing $\mathbf{n}$.

Intuitively, the value 0 for a node $\mathbf{n}$ with respect to a branch B means that $\mathbf{n}$ can be expanded on B. For the sake of simplicity, we will often assume the interval $[c_i, c_j]$ to consist of the elements $c_i < c_{i+1} < \cdots < c_{j-1} < c_j$, and sometimes, with a little abuse of notation, we will write $\mathbb{C} = \{c_i < c_k, c_m < c_j, \ldots\}$.

Definition 2. A *decorated tree* is a tree in which every node has a decoration $\nu(\mathbf{n})$.

For every decorated tree, we also use a *global flag function* u acting on pairs (*node, branch through that node*), and defined as $u(\mathbf{n}, B) \triangleq u_{\mathbf{n}}(B)$. Sometimes, for convenience, we will include in the decoration of the nodes the global flag function instead of the local ones. For any branch B in a decorated tree, we denote by $\mathbb{C}_B$ the (partially) ordered set in the decoration of the leaf of B, and for any node $\mathbf{n}$ in a decorated tree, we denote by $\Phi(\mathbf{n})$ the formula in its decoration. If B is a branch, then $B \cdot \mathbf{n}$ denotes the result of the expansion of B with the node $\mathbf{n}$ (addition of an edge connecting the leaf of B to $\mathbf{n}$). Similarly, $B \cdot \mathbf{n}_1 \mid \ldots \mid \mathbf{n}_k$ denotes the result of the expansion of B with k immediate successor nodes $\mathbf{n}_1, \ldots, \mathbf{n}_k$ (which produces k branches extending B). A tableau for BCDT$^+$ will be defined as a special decorated tree. We note again that $\mathbb{C}$ remains finite throughout the construction of the tableau.

Definition 3. Given a decorated tree $\mathcal{T}$, a branch B in $\mathcal{T}$, and a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\phi, [c_i, c_j]), \mathbb{C}, u)$, with $u(\mathbf{n}, B) = 0$, the *branch-expansion rule* for B and $\mathbf{n}$ is defined as follows (in all the considered cases, $u(\mathbf{n}', B') = 0$ for all new pairs $(\mathbf{n}', B')$ of nodes and branches):

- If $\phi = \neg\neg\psi$, then expand the branch to $B \cdot \mathbf{n}_0$, with $\nu(\mathbf{n}_0) = ((\psi, [c_i, c_j]), \mathbb{C}_B, u)$;

- If $\phi = \psi_0 \wedge \psi_1$, then expand the branch to $B \cdot \mathbf{n}_0 \cdot \mathbf{n}_1$, with $\nu(\mathbf{n}_0) = ((\psi_0, [c_i, c_j]), \mathbb{C}_B, u)$ and $\nu(\mathbf{n}_1) = ((\psi_1, [c_i, c_j]), \mathbb{C}_B, u)$;
- If $\phi = \neg(\psi_0 \wedge \psi_1)$, then expand the branch to $B \cdot \mathbf{n}_0 | \mathbf{n}_1$, with $\nu(\mathbf{n}_0) = ((\neg\psi_0, [c_i, c_j]), \mathbb{C}_B, u)$ and $\nu(\mathbf{n}_1) = ((\neg\psi_1, [c_i, c_j]), \mathbb{C}_B, u)$;
- If $\phi = \neg(\psi_0 C \psi_1)$ and there exists $c_k \in \mathbb{C}_B$, with $c_k \in [c_i, c_j]$, which has not been used yet to expand the node $\mathbf{n}$ on B, then take the least such c_k and expand the branch to $B \cdot \mathbf{n}_0 | \mathbf{n}_1$, with $\nu(\mathbf{n}_0) = ((\neg\psi_0, [c_i, c_k]), \mathbb{C}_B, u)$ and $\nu(\mathbf{n}_1) = ((\neg\psi_1, [c_k, c_j]), \mathbb{C}_B, u)$;
- If $\phi = \neg(\psi_0 D \psi_1)$, c is a minimal element of $\mathbb{C}_B$ such that $c \leqslant c_i$, and there exists $c' \in [c, c_i]$, which has not been used yet to expand the node $\mathbf{n}$ on B, then take the least such c' and expand the branch to $B \cdot \mathbf{n}_0 | \mathbf{n}_1$, with $\nu(\mathbf{n}_0) = ((\neg\psi_0, [c', c_i]), \mathbb{C}_B, u)$ and $\nu(\mathbf{n}_1) = ((\neg\psi_1, [c', c_j]), \mathbb{C}_B, u)$;
- If $\phi = \neg(\psi_0 T \psi_1)$, c is a maximal element of $\mathbb{C}_B$ such that $c_j \leqslant c$, and there exists $c' \in [c_j, c]$ which has not been used yet to expand the node $\mathbf{n}$ on B, then take the greatest such c' and expand the branch to $B \cdot \mathbf{n}_0 | \mathbf{n}_1$, with $\nu(\mathbf{n}_0) = ((\neg\psi_0, [c_j, c']), \mathbb{C}_B, u)$ and $\nu(\mathbf{n}_1) = ((\neg\psi_1, [c_i, c']), \mathbb{C}_B, u)$;
- If $\phi = (\psi_0 C \psi_1)$, then expand the branch to $B \cdot (\mathbf{n}_i \cdot \mathbf{m}_i) | \ldots | (\mathbf{n}_j \cdot \mathbf{m}_j) | (\mathbf{n}'_i \cdot \mathbf{m}'_i) | \ldots | (\mathbf{n}'_{j-1} \cdot \mathbf{m}'_{j-1})$, where:
  (1) for all $c_k \in [c_i, c_j]$, $\nu(\mathbf{n}_k) = ((\psi_0, [c_i, c_k]), \mathbb{C}_B, u)$ and $\nu(\mathbf{m}_k) = ((\psi_1, [c_k, c_j]), \mathbb{C}_B, u)$;
  (2) for all $i \leqslant k \leqslant j - 1$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c between c_k and c_{k+1} in $[c_i, c_j]$, $\nu(\mathbf{n}'_k) = ((\psi_0, [c_i, c]), \mathbb{C}_k, u)$, and $\nu(\mathbf{m}'_k) = ((\psi_1, [c, c_j]), \mathbb{C}_k, u)$;
- If $\phi = (\psi_0 D \psi_1)$, then repeatedly expand the current branch, once for each minimal element c (where $[c, c_i] = \{c = c_0 < c_1 < \cdots < c_i\}$), by adding the decorated subtree $(\mathbf{n}_0 \cdot \mathbf{m}_0) | \ldots | (\mathbf{n}_i \cdot \mathbf{m}_i) | (\mathbf{n}'_1 \cdot \mathbf{m}'_1) | \ldots | (\mathbf{n}'_i \cdot \mathbf{m}'_i) | (\mathbf{n}''_0 \cdot \mathbf{m}''_0) | \ldots | (\mathbf{n}''_i \cdot \mathbf{m}''_i)$ to its leaf, where:
  (1) for all $c_k \in [c_0, c_i]$, $\nu(\mathbf{n}_k) = ((\psi_0, [c_k, c_i]), \mathbb{C}_B, u)$ and $\nu(\mathbf{m}_k) = ((\psi_1, [c_k, c_j]), \mathbb{C}_B, u)$;
  (2) for all $1 \leqslant k \leqslant i$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c' between c_{k-1} and c_k in $[c_0, c_i]$, $\nu(\mathbf{n}'_k) = ((\psi_0, [c', c_i]), \mathbb{C}_k, u)$, and $\nu(\mathbf{m}'_k) = ((\psi_1, [c', c_j]), \mathbb{C}_k, u)$;
  (3) for all $0 \leqslant k \leqslant i$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c' in $\mathbb{C}_B$, with $c' < c_k$, which is incomparable with all existing predecessors of c_k, $\nu(\mathbf{n}''_k) = ((\psi_0, [c', c_i]), \mathbb{C}_k, u)$, and $\nu(\mathbf{m}''_k) = ((\psi_1, [c', c_j]), \mathbb{C}_k, u)$;
- If $\phi = (\psi_0 T \psi_1)$, then repeatedly expand the current branch, once for each maximal element c (where $[c_j, c] = \{c_j < c_{j+1} < \cdots < c_n = c\}$), by adding the decorated subtree $(\mathbf{n}_j \cdot \mathbf{m}_j) | \ldots | (\mathbf{n}_n \cdot \mathbf{m}_n) | (\mathbf{n}'_j \cdot \mathbf{m}'_j) | \ldots | (\mathbf{n}'_{n-1} \cdot \mathbf{m}'_{n-1}) | (\mathbf{n}''_j \cdot \mathbf{m}''_j) | \ldots | (\mathbf{n}''_n \cdot \mathbf{m}''_n)$ to its leaf, where:
  (1) for all $c_k \in [c_j, c]$, $\nu(\mathbf{n}_k) = ((\psi_0, [c_j, c_k]), \mathbb{C}_B, u)$ and $\nu(\mathbf{m}_k) = ((\psi_1, [c_i, c_k]), \mathbb{C}_B, u)$;
  (2) for all $j \leqslant k \leqslant n - 1$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c' between c_k and c_{k+1} in $[c_j, c_n]$, $\nu(\mathbf{n}'_k) = ((\psi_0, [c_j, c']), \mathbb{C}_k, u)$, and $\nu(\mathbf{m}'_k) = ((\psi_1, [c_i, c']), \mathbb{C}_k, u)$;
  (3) for all $j \leqslant k \leqslant n$, let $\mathbb{C}_k$ be the partial ordering obtained by inserting a new element c' in $\mathbb{C}_B$, with $c_k < c'$, which is incomparable with all existing successors of c_k, $\nu(\mathbf{n}''_k) = ((\psi_0, [c_j, c']), \mathbb{C}_k, u)$, and $\nu(\mathbf{m}''_k) = ((\psi_1, [c_i, c']), \mathbb{C}_k, u)$.

Finally, for any node $\mathbf{m}$ ($\neq \mathbf{n}$) in B and any branch B' extending B, let $u(\mathbf{m}, B') = u(\mathbf{m}, B)$, and for any branch B' extending B, $u(\mathbf{n}, B') = 1$, unless $\phi = \neg(\psi_0 C \psi_1)$, $\phi = \neg(\psi_0 D \psi_1)$, or $\phi = \neg(\psi_0 T \psi_1)$ (in such cases $u(\mathbf{n}, B') = 0$).

Let us briefly explain the expansion rules for $\psi_0 C \psi_1$ and $\neg(\psi_0 C \psi_1)$ (similar considerations hold for the other temporal operators). The rule for the existential formula $\psi_0 C \psi_1$ deals with the two possible cases: either there exists $c_k \in \mathbb{C}_B$ such that $c_i \leqslant c_k \leqslant c_j$, ψ_0 holds over $[c_i, c_k]$, and ψ_1 holds over $[c_k, c_j]$ ($j - i + 1$ cases) or such an element c_k must be added to $\mathbb{C}_B$ ($j - i$ cases). The universal formula $\neg(\psi_0 C \psi_1)$ states that, for all $c_i \leqslant c_k \leqslant c_j$, ψ_0 does not hold over $[c_j, c_k]$ or ψ_1 does not hold over $[c_k, c_j]$ ($j - i + 1$ cases). As a matter of fact, the expansion rule imposes such a condition for a single element c_k in $\mathbb{C}_B$ (the least $c_i \leqslant c_k \leqslant c_j$ which has not been used yet), and it does not change the flag (which remains equal to 0). In this way, all elements will be eventually taken into consideration, including those elements in between c_i and c_j that will be added to $\mathbb{C}_B$ in some subsequent steps of the tableau construction.

Let us define now the notions of open and closed branch. We say that a node $\mathbf{n}$ in a decorated tree $\mathcal{T}$ is *available on a branch* B it belongs to if and only if $u(\mathbf{n}, B) = 0$. The branch-expansion rule is *applicable* to a node $\mathbf{n}$ on a branch B if the node is available on B and the application of the rule generates at least one successor node with a new labeled formula. This second condition is needed to avoid looping of the application of the rule on formulas $\neg(\psi_0 C \psi_1)$, $\neg(\psi_0 D \psi_1)$, and $\neg(\psi_0 T \psi_1)$.

Definition 4. A branch B is *closed* if some of the following conditions holds:

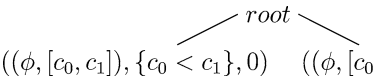
(1) there are two nodes $\mathbf{n}, \mathbf{n}' \in B$ such that $\nu(\mathbf{n}) = ((\psi, [c_i, c_j]), \mathbb{C}, u)$ and $\nu(\mathbf{n}') = ((\neg\psi, [c_i, c_j]), \mathbb{C}', u)$ for some formula ψ and $c_i, c_j \in C \cap C'$;
(2) there is a node $\mathbf{n}$ such that $\nu(\mathbf{n}) = ((\pi, [c_i, c_j]), \mathbb{C}, u)$ and $c_i \neq c_j$;
(3) there is a node $\mathbf{n}$ such that $\nu(\mathbf{n}) = ((\neg\pi, [c_i, c_j]), \mathbb{C}, u)$ and $c_i = c_j$.

If none of the above conditions hold, the branch is *open*.

Definition 5. The *branch-expansion strategy* for a branch B in a decorated tree $\mathcal{T}$ is defined as follows:

(1) apply the branch-expansion rule to a branch B only if it is open;
(2) if B is open, apply the branch-expansion rule to the first available node one encounters moving from the root to the leaf of B to which the branch-expansion rule is applicable (if any).

Definition 6. An *initial tableau* for a given formula $\phi \in \mathrm{BCDT}^+$ is the following finite decorated tree $\mathcal{T}$:



where the value of u is 0. A *tableau* for a given formula $\phi \in \mathrm{BCDT}^+$ is any finite decorated tree isomorphic to a finite decorated tree $\mathcal{T}$ obtained by expanding the initial tableau for ϕ through successive applications of the branch-expansion strategy to the existing branches.

It is easy to show that if $\phi \in \mathrm{BCDT}^+$, $\mathcal{T}$ is a tableau for ϕ, $\mathbf{n} \in \mathcal{T}$, and $\mathbb{C}$ is the ordered set in the decoration of $\mathbf{n}$, then $\mathbb{C}$ has the linear interval property.

Definition 7. A tableau for BCDT^+ is *closed* if and only if every branch in it is closed, otherwise it is *open*.

We conclude the section by giving some examples of the application of the proposed method (for the sake of readability, we omit some minor details in the figures). As a first example, let ϕ be the unsatisfiable formula $pT\neg(\top Cp)$. A closed tableau for ϕ is given in Fig. 2. As a second example, let ϕ be the formula $\neg\pi \wedge \neg(\neg((p \wedge \neg\pi)T\neg p)T\neg((p \wedge \neg\pi)T\neg p))$, which is satisfiable, but it only admits infinite models. Hence, all its tableaux include at least one open branch. A tableau (to be further expanded) for ϕ is given in Fig. 3. Finally, let ϕ be the formula $p \leftrightarrow pT\pi$, which is one of the CDT axioms given in [32]. Such an axiomatic system is (claimed to be) sound and complete for the class of all linear (non-strict) interval structures. From this, it follows that the negation of the formula $(p \rightarrow pT\pi) \wedge \mathit{lin}$ (cf. Section 3) should be unsatisfiable in the class of all (non-strict) interval structures. However, the open tableau for $\neg(p \rightarrow pT\pi)$ shown in Fig. 4 proves that this is not the case (as matter of fact, to remedy this it suffices to substitute $pT\pi \rightarrow p$ for $p \leftrightarrow pT\pi$).

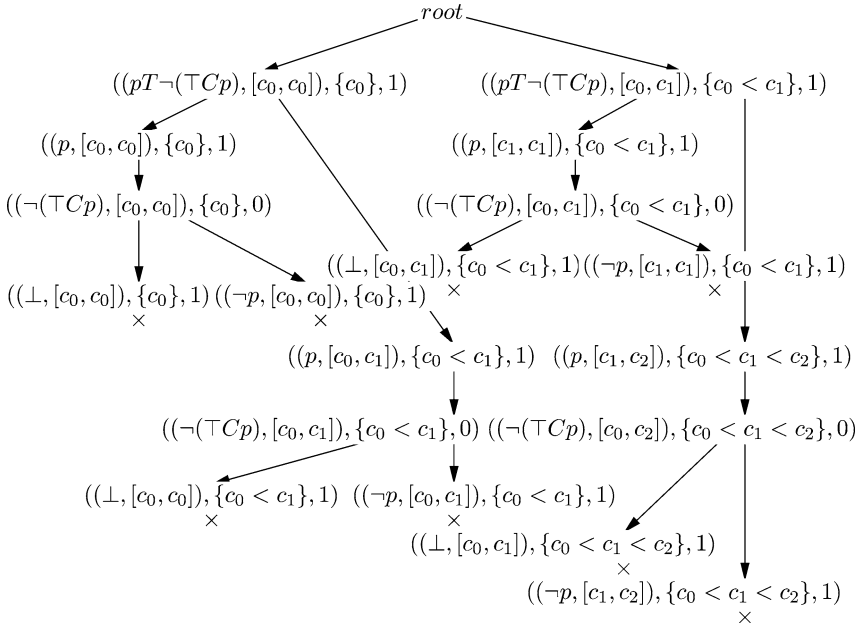


Fig. 2. A closed tableau for the formula $pT\neg(\top Cp)$.

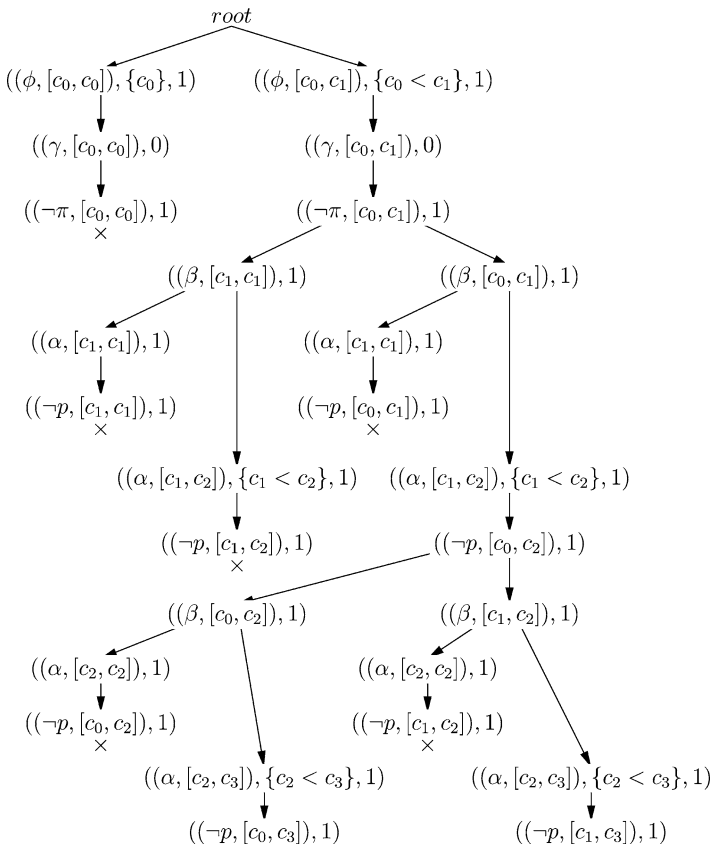


Fig. 3. A tableau (to be further expanded) for the formula $\neg\pi\wedge\neg(\neg((p\wedge\neg\pi)T\neg p)T\neg((p\wedge\neg\pi)T\neg p))$, where α, β, and γ stand for $p\wedge\neg\pi$, $\alpha T\neg p$, and $\neg(\neg\beta T\neg\beta)$, respectively.

4.1. Soundness and completeness

Definition 8. Given a set S of labeled formulas with labels in $\mathbb{C}$, we say that S is *satisfiable over* $\mathbb{C}$ if there exists a non-strict model $\mathbf{M}^+=\langle\mathbb{D},\mathbb{I}(D)^+,V\rangle$ such that $\mathbb{D}$ is an extension of $\mathbb{C}$ and $\mathbf{M}^+,[c_i,c_j]\Vdash\psi$ for all $(\psi,[c_i,c_j])\in S$.

If S contains only one labeled formula, the notion of satisfiability of a (labeled) formula over $\mathbb{C}$ is equivalent to the notion of satisfiability given in Section 2.

Theorem 1 *(Soundness). If $\phi\in$ BCDT$^+$ and a tableau $\mathcal{T}$ for ϕ is closed, then ϕ is not satisfiable.*

Proof. We will prove by induction on the height h of a node $\mathbf{n}$ in the tableau $\mathcal{T}$ the following claim: *if every branch including* $\mathbf{n}$ *is closed, then the set* $S(\mathbf{n})$ *of all labeled formulas*

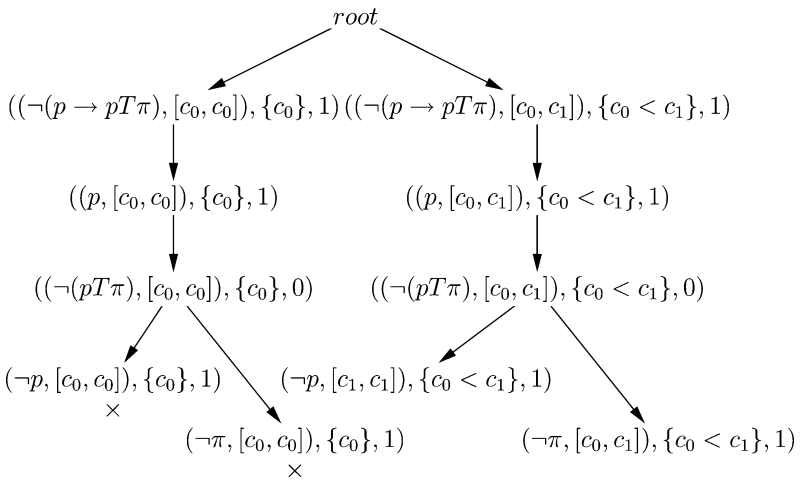


Fig. 4. An open tableau for the formula $\neg(p \to pT\pi)$.

in the decorations of the nodes between $\mathbf{n}$ *and the root is not satisfiable over* $\mathbb{C}$*, where* $\mathbb{C}$ *is the interval structure in the decoration of* $\mathbf{n}$.

If $h = 0$, then $\mathbf{n}$ is a leaf and the unique branch B containing $\mathbf{n}$ is closed. Then, either $S(\mathbf{n})$ contains both the labeled formulas $(\psi, [c_k, c_l])$ and $(\neg\psi, [c_k, c_l])$ for some BCDT$^+$-formula ψ and $c_k, c_l \in \mathbb{C}$, or the labeled formula $(\pi, [c_k, c_l])$ and $c_k \neq c_l$, or the labeled formula $(\neg\pi, [c_k, c_l])$ and $c_k = c_l$. Take any model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(D)^+, V\rangle$ where $\mathbb{D}$ is an extension of $\mathbb{C}$. In the first case, clearly $\mathbf{M}^+, [c_k, c_l] \Vdash \psi$ if and only if $\mathbf{M}^+, [c_k, c_l] \nVdash \neg\psi$. In the second (resp., third) case, $\mathbf{M}^+, [c_k, c_l] \Vdash \pi$ (resp., $\neg\pi$) if and only if $c_k = c_l$ (resp., $c_k \neq c_l$). Hence, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$. Otherwise, suppose $h > 0$. Then either $\mathbf{n}$ has been generated as one of the successors, but not the last one, when applying the branch-expansion rule in $\wedge$, C, D, T, $\neg C, \neg D$, or $\neg T$ cases, or the branch-expansion rule has been applied to some labeled formula $(\psi, [c_i, c_j]) \in S(\mathbf{n}) - \{\Phi(\mathbf{n})\}$ to extend the branch at $\mathbf{n}$. We deal with the latter case. The former can be dealt with in the same way. Let $\mathbb{C} = \{c_1, \ldots, c_n\}$, be the interval structure from the decoration of $\mathbf{n}$. Notice that every branch passing through any successor of $\mathbf{n}$ must be closed, so the inductive hypothesis applies to all successors of $\mathbf{n}$. We consider the possible cases for the branch-expansion rule applied at $\mathbf{n}$:

- Let $\psi = \neg\neg\xi$. Then there exists $\mathbf{n}_0$ such that $\nu(\mathbf{n}_0) = ((\xi, [c_i, c_j]), \mathbb{C}, u)$ and $\mathbf{n}_0$ is a successor of $\mathbf{n}$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ is closed. By the inductive hypothesis, $S(\mathbf{n}_0)$ is not satisfiable over $\mathbb{C}$ (since $\mathbf{n}_0 \prec \mathbf{n}$). Since ξ_0 and $\neg\neg\xi_0$ are equivalent, $S(\mathbf{n})$ cannot be satisfiable over $\mathbb{C}$;
- Let $\psi = \xi_0 \wedge \xi_1$. Then there are two nodes $\mathbf{n}_0 \in B$ and $\mathbf{n}_1 \in B$ such that $\nu(\mathbf{n}_0) = ((\xi_0, [c_i, c_j]), \mathbb{C}, u)$, $\nu(\mathbf{n}_1) = ((\xi_1, [c_i, c_j]), \mathbb{C}, u)$, and, without loss of generality, $\mathbf{n}_0$ is the successor of $\mathbf{n}$ and $\mathbf{n}_1$ is the successor of $\mathbf{n}_0$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_1$ is closed. By the inductive hypothesis, $S(\mathbf{n}_1)$ is not satisfiable over $\mathbb{C}$ since $\mathbf{n}_1 \prec \mathbf{n}$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$

must, in particular, satisfy $(\xi_0 \wedge \xi_1, [c_i, c_j])$, and hence $(\xi_0, [c_i, c_j])$ and $(\xi_1, [c_i, c_j])$, it follows that $S(\mathbf{n})$, $S(\mathbf{n}_0)$, and $S(\mathbf{n}_1)$ are equi-satisfiable over $\mathbb{C}$. Therefore, $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- Let $\psi = \neg(\xi_1 \wedge \xi_2)$. Then there exist two successor nodes $\mathbf{n}_0$ and $\mathbf{n}_1$ of $\mathbf{n}$ such that $\nu(\mathbf{n}_0) = ((\xi_0, [c_i, c_j]), \mathbb{C}, u_0)$, $\nu(\mathbf{n}_1) = ((\xi_1, [c_i, c_j]), \mathbb{C}, u_1)$, $\mathbf{n}_0, \mathbf{n}_1 \prec \mathbf{n}$. Since every branch containing $\mathbf{n}$ is closed, then every branch containing $\mathbf{n}_0$ and every branch containing $\mathbf{n}_1$ is closed. By the inductive hypothesis $S(\mathbf{n}_0)$ and $S(\mathbf{n}_1)$ are not satisfiable over $\mathbb{C}$. Since every model over $\mathbb{C}$ satisfying $S(\mathbf{n})$ must also satisfy $(\xi_0, [c_i, c_j])$ or $(\xi_1, [c_i, c_j])$, it follows that $S(\mathbf{n})$ cannot be satisfiable over $\mathbb{C}$;
- Let $\psi = \neg(\xi_0 C \xi_0)$. Suppose that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$. Since $(\neg(\xi_0 C \xi_1), [c_i, c_j]) \in S(\mathbf{n})$, there is a model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(D)^+, V \rangle$ such that $\mathbb{D}$ is an extension of $\mathbb{C}$ and $\mathbf{M}^+, [c_i, c_j] \Vdash \neg(\xi_0 C \xi_1)$. So, for every c_k such that $c_i \leqslant c_k \leqslant c_j$, we have that $\mathbf{M}^+, [c_i, c_k] \Vdash \neg\xi_0$ or $\mathbf{M}^+, [c_k, c_j] \Vdash \neg\xi_1$. By construction, the two immediate successors of $\mathbf{n}$ are $\mathbf{n}_1$ and $\mathbf{n}_2$ such that, for an element c_k with $c_i \leqslant c_k \leqslant c_j$, $(\neg\xi_0, [c_i, c_k])$ is in the decoration of $\mathbf{n}_0$ and $(\neg\xi_1, [c_k, c_j])$ is in the decoration of $\mathbf{n}_1$. By inductive hypothesis, since $\mathbf{n}_1, \mathbf{n}_2 \prec \mathbf{n}$, $S(\mathbf{n}_1)$ and $S(\mathbf{n}_2)$ are not satisfiable over $\mathbb{C}$. Thus, such a model $\mathbf{M}^+$ cannot exist, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;
- The cases $\psi = \neg(\xi_0 D \xi_1)$ and $\psi = \neg(\xi_0 T \xi_1)$ are analogous;
- Let $\psi = \xi_0 C \xi_1$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(D)^+, V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that $\mathbf{M}^+, [c_i, c_j] \Vdash \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathbf{M}^+, [c_i, d] \Vdash \xi_0$ and $\mathbf{M}^+, [d, c_j] \Vdash \xi_1$ for some $c_i \leqslant d \leqslant c_j$. Consider two cases:
  (1) If $d \in \mathbb{C}$, then $d = c_m$ for some $c_i \leqslant c_m \leqslant c_j$. But among the successors of $\mathbf{n}$ there are two nodes $\mathbf{n}_m, \mathbf{m}_m$ where $\nu(\mathbf{n}_m) = ((\xi_0, [c_i, c_m]), \mathbb{C}, u)$ and $\nu(\mathbf{m}_m) = ((\xi_1, [c_m, c_j]), \mathbb{C}, u)$, and since $\mathbf{n}_m, \mathbf{m}_m \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n}_m \prec \mathbf{m}_m$), by the inductive hypothesis $S(\mathbf{n}_m) = S(\mathbf{n}) \cup \{(\xi_0, [c_i, c_m]), (\xi_1, [c_m, c_j])\}$ is not satisfiable over $\mathbb{C}$, which is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;
  (2) If $d \notin \mathbb{C}$, then there is an m such that $i \leqslant m \leqslant j-1$ and $c_m < d < c_{m+1}$. Hence, there are two successors $\mathbf{n}'_m, \mathbf{m}'_m$ of $\mathbf{n}$ such that $\nu(\mathbf{n}'_m) = ((\xi_0, [c_i, d]), \mathbb{C} \cup \{d\}, u)$, $\nu(\mathbf{m}'_m) = ((\xi_1, [d, c_j]), \mathbb{C} \cup \{d\}, u)$, and since $\mathbf{n}'_m, \mathbf{m}'_m \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n}'_m \prec \mathbf{m}'_m$), by the inductive hypothesis $S(\mathbf{n}'_m) = S(\mathbf{n}) \cup \{(\xi_0, [c_i, d]), (\xi_1, [d, c_j])\}$ is not satisfiable over $\mathbb{C} \cup \{d\}$ which, again, is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;
- Let $\psi = \xi_0 D \xi_1$. Assuming that $S(\mathbf{n})$ is satisfiable over $\mathbb{C}$, there is a model $\mathbf{M}^+ = \langle \mathbb{D}, \mathbb{I}(D)^+, V \rangle$, where $\mathbb{D}$ is an extension of $\mathbb{C}$, such that $\mathbf{M}^+, [c_i, c_j] \Vdash \theta$ for all $(\theta, [c_i, c_j]) \in S(\mathbf{n})$. In particular, $\mathbf{M}^+, [d, c_i] \Vdash \xi_0$ and $\mathbf{M}^+, [d, c_j] \Vdash \xi_1$ for some $d \leqslant c_i$. Consider 3 cases:
  (1) If $d \in \mathbb{C}$, then $d = c_m$ for some $c_m \leqslant c_i$. But between the successors of $\mathbf{n}$ there are two nodes $\mathbf{n}_m, \mathbf{m}_m$ where $\nu(\mathbf{n}_m) = ((\xi_0, [c_m, c_i]), \mathbb{C}, u)$ and $\nu(\mathbf{m}_m) = ((\xi_1, [c_m, c_j]), \mathbb{C}, u)$, and since $\mathbf{n}_m, \mathbf{m}_m \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n}_m \prec \mathbf{m}_m$), by the inductive hypothesis $S(\mathbf{n}_m) = S(\mathbf{n}) \cup \{(\xi_0, [c_m, c_i]), (\xi_1, [c_m, c_j])\}$ is not satisfiable over $\mathbb{C}$, which is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

(2) If $d \notin \mathbb{C}$ and there exist a minimal element $c \in \mathbb{C}$ and an index m such that $c_m, c_{m+1} \in [c, c_i]$ and $c_m < d < c_{m+1}$, then there are two successors $\mathbf{n}'_m, \mathbf{m}'_m$ of $\mathbf{n}$ such that $\nu(\mathbf{n}'_m) = ((\xi_0, [d, c_i]), \mathbb{C} \cup \{d\}, u)$ and $\nu(\mathbf{m}'_m) = ((\xi_1, [d, c_j]), \mathbb{C} \cup \{d\}, u)$, and since $\mathbf{n}'_m, \mathbf{m}'_m \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n}'_m \prec \mathbf{m}'_m$), by the inductive hypothesis $S(\mathbf{n}'_m) = S(\mathbf{n}) \cup \{(\xi_0, [d, c_i]), (\xi_1, [d, c_j])\}$ is not satisfiable over $\mathbb{C} \cup \{d\}$ which, again, is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

(3) If $d \notin \mathbb{C}$ and there exist a minimal element $c \in \mathbb{C}$ and an index m such that $c_{m+1} \in [c, c_i]$, $d < c_{m+1}$, and d is not comparable with all predecessors of c_{m+1}, then, again, there are two successor nodes $\mathbf{n}''_m, \mathbf{m}''_m$ of $\mathbf{n}$ such that $\nu(\mathbf{n}''_m) = ((\xi_0, [d, c_i]), \mathbb{C} \cup \{d\}, u)$ and $\nu(\mathbf{m}''_m) = ((\xi_1, [d, c_j]), \mathbb{C} \cup \{d\}, u)$, and since $\mathbf{n}''_m$, $\mathbf{m}''_m \prec \mathbf{n}$ (without loss of generality, suppose $\mathbf{n}''_m \prec \mathbf{m}''_m$), by the inductive hypothesis $S(\mathbf{n}''_m) = S(\mathbf{n}) \cup \{(\xi_0, [d, c_i]), (\xi_1, [d, c_j])\}$ is not satisfiable over $\mathbb{C} \cup \{d\}$ which, again, is a contradiction, and $S(\mathbf{n})$ is not satisfiable over $\mathbb{C}$;

- The case of $\psi = \xi_0 T \xi_1$ is similar. □

Definition 9. If $\mathcal{T}_0$ is the three-node tableau built up from a root with void decoration and two leaves decorated respectively by $((\phi, [c_b, c_e]), \{c_b < c_e\}, 0)$ and $((\phi, [c_b, c_b]), \{c_b\}, 0)$ for a given BCDT$^+$-formula ϕ, the *limit tableau* $\overline{\mathcal{T}}$ for ϕ is the (possibly infinite) decorated tree obtained as follows. First, for all i, $\mathcal{T}_{i+1}$ is the tableau obtained by the simultaneous application of the branch-expansion strategy to every branch in $\mathcal{T}_i$. Then, we ignore all flags from the decorations of the nodes in every $\mathcal{T}_i$. Thus, we obtain a chain by inclusion of decorated trees: $\mathcal{T}_1 \subseteq \mathcal{T}_2 \subseteq \cdots$, and we define $\overline{\mathcal{T}} = \bigcup_{i=0}^{\infty} \mathcal{T}_i$.

Notice that the chain above may stabilize at some $\mathcal{T}_i$ if it closes, or if the branch-expansion rule is not applicable to any of its branches. If $\overline{\mathcal{T}}$ is a limit tableau, we associate with each branch B in $\overline{\mathcal{T}}$ the interval structure $\mathbb{C}_B = \bigcup_{i=0}^{\infty} \mathbb{C}_{B_i}$, where, for all i, $\mathbb{C}_{B_i}$ is the interval structure from the decoration of the leaf of the (sub-)branch B_i of B in $\mathcal{T}_i$. The definitions of closed and open branches readily apply to $\overline{\mathcal{T}}$.

Definition 10. A branch in a (limit) tableau is *saturated* if there are no nodes on that branch to which the branch-expansion rule is applicable on the branch. A (limit) tableau is *saturated* if every open branch in it is saturated.

Now we will show that the set of all labeled formulas on an open branch in a limit tableau has the saturation properties of a Hintikka set in first-order logic.

Lemma 2. *Every limit tableau is saturated.*

Proof. Given a node $\mathbf{n}$ in a limit tableau $\overline{\mathcal{T}}$, we denote by $d(\mathbf{n})$ the distance (number of edges) between $\mathbf{n}$ and the root of $\overline{\mathcal{T}}$. Now, given a branch B in $\overline{\mathcal{T}}$, we will prove by induction on $d(\mathbf{n})$ that after every step of the expansion of that branch at which the branch-expansion rule becomes applicable to $\mathbf{n}$ (because $\mathbf{n}$ has just been introduced, or because a new point has been introduced in the interval structure on B) that rule is subsequently applied on B to that node.

Suppose the inductive hypothesis holds for all nodes with distance to the root less than l. Let $d(\mathbf{n}) = l$ and the branch-expansion rule has become applicable to $\mathbf{n}$. If there are no nodes between the root (incl. the root) and $\mathbf{n}$ (excl. $\mathbf{n}$) to which the branch-expansion rule is applicable at that moment, the next application of the branch-expansion rule on B is to $\mathbf{n}$. Otherwise, consider the closest-to-$\mathbf{n}$ node $\mathbf{n}^*$ between the root and $\mathbf{n}$ to which the branch-expansion rule is applicable or will become applicable on B at least once thereafter. (Such a node exists because there are only finitely many nodes between $\mathbf{n}$ and the root.) Since $d(\mathbf{n}^*) < d(\mathbf{n})$, by the inductive hypothesis the branch-expansion rule has been subsequently applied to $\mathbf{n}^*$. Then the next application of the branch-expansion rule on B must have been to $\mathbf{n}$ and that completes the induction. Now, assuming that a branch in a limit tableau is not saturated, consider the closest-to-the-root node $\mathbf{n}$ on that branch B to which the branch-expansion rule is applicable on that branch. If $\Phi(\mathbf{n})$ is none of the cases $\neg C, \neg D$, and $\neg T$, then the branch-expansion rule has become applicable to $\mathbf{n}$ at the step when $\mathbf{n}$ is introduced, and by the claim above, it has been subsequently applied, at which moment the node has become unavailable thereafter, which contradicts the assumption. Suppose that $\Phi(\mathbf{n}) = \neg(\psi_0 C \psi_1)$. Then an application of the rule on B would create two successors with labels $(\neg\psi_0, [c_i, c])$ and $(\neg\psi_1, [c, c_j])$, at least one of them new on B. But c_i, c_j, c have already been introduced at some (finite) step of the construction of B and at the first step when the three of them, as well as $\mathbf{n}$, have appeared on the branch, the branch-expansion rule has become applicable to $\mathbf{n}$, hence is has been subsequently applied on B and that application must have introduced the labels $(\psi_0, [c_i, c])$ and $(\psi_1, [c, c_j])$ on B, which again contradicts the assumption. The same holds if $\Phi(\mathbf{n}) = \neg(\psi_0 D \psi_1)$ or $\Phi(\mathbf{n}) = \neg(\psi_0 D \psi_1)$. □

Corollary 3. *Let ϕ be a* BCDT$^+$*-formula and $\overline{T}$ be the limit tableau for ϕ. For every open branch B in $\overline{T}$, the following closure properties hold*:

- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\neg\neg\psi, [c_i, c_j]), \mathbb{C}, u)$, then there is a node $\mathbf{n}_0 \in B$ such that $\nu(\mathbf{n}_0) = ((\psi, [c_i, c_j]), \mathbb{C}, u_0)$;*
- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\psi_0 \wedge \psi_1, [c_i, c_j]), \mathbb{C}, u)$, then there is a node $\mathbf{n}_0 \in B$ such that $\nu(\mathbf{n}_0) = ((\psi_0, [c_i, c_j]), \mathbb{C}, u_0)$ and a node $\mathbf{n}_1 \in B$ such that $\nu(\mathbf{n}_1) = ((\psi_1, [c_i, c_j]), \mathbb{C}, u_1)$;*
- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\neg(\psi_0 \wedge \psi_1), [c_i, c_j]), \mathbb{C}, u)$, then there is a node $\mathbf{n}_0 \in B$ such that $\nu(\mathbf{n}_0) = ((\neg\psi_0, [c_i, c_j]), \mathbb{C}, u_0)$ or a node $\mathbf{n}_1 \in B$ such that $\nu(\mathbf{n}_1) = ((\neg\psi_1, [c_i, c_j]), \mathbb{C}, u_1)$;*
- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\psi_0 C \psi_1, [c_i, c_j]), \mathbb{C}, u)$, then, for some $c \in \mathbb{C}_B$ such that $c_i \leqslant c \leqslant c_j$ there are two nodes $\mathbf{n}', m' \in B$ such that $\nu(\mathbf{n}') = ((\psi_0, [c_i, c]), \mathbb{C}', u')$ and $\nu(\mathbf{m}') = ((\psi_1, [c, c_j]), \mathbb{C}', u')$;*
- *Similarly for every node $\mathbf{n}$ with $\Phi(\mathbf{n}) = \psi_0 D \psi_1$ or $\Phi(\mathbf{n}) = \psi_0 T \psi_1$;*
- *If there is a node $\mathbf{n} \in B$ such that $\nu(\mathbf{n}) = ((\neg(\psi_0 C \psi_1), [c_i, c_j]), \mathbb{C}, u)$, then for all $c \in \mathbb{C}_B$ such that $c_i \leqslant c \leqslant c_j$, there is a node $\mathbf{n}' \in B$ such that $\nu(\mathbf{n}') = ((\neg\psi_0, [c_i, c]), \mathbb{C}', u')$ or a node $\mathbf{m}' \in B$ such that $\nu(\mathbf{m}') = ((\neg\psi_1, [c, c_j]), \mathbb{C}', u')$;*
- *Similarly for every node $\mathbf{n}$ with $\Phi(\mathbf{n}) = \neg(\psi_0 D \psi_1)$ or $\Phi(\mathbf{m}) = \neg(\psi_0 T \psi_1)$.*

Lemma 4. *If the limit tableau for some formula $\phi \in \mathrm{BCDT}^+$ is closed, then some finite tableau for ϕ is closed.*

Proof. Suppose the limit tableau for ϕ is closed. Then every branch closes at some finite step of the construction and then remains finite. Since the branch-expansion rule always produces finitely many successors, every finite tableau is finitely branching, and hence so is the limit tableau. Then, by König's lemma, the limit tableau, being a finitely branching tree with no infinite branches, must be finite, hence its construction stabilizes at some finite stage. At that stage a closed tableau for ϕ is constructed. □

Theorem 5 *(Completeness). Let $\phi \in \mathrm{BCDT}^+$ be a valid formula. Then there is a closed tableau for $\neg\phi$.*

Proof. We will show that the limit tableau $\overline{\mathcal{T}}$ for $\neg\phi$ is closed, whence the claim follows by the previous lemma.

By contraposition, suppose that $\overline{\mathcal{T}}$ has an open branch B. Let $\mathbb{C}_B$ be the interval structure associated with B and $S(B)$ be the set of all labeled formulas on B. Consider the model $\mathbf{M}^+ = \langle \mathbb{C}_B, V \rangle$ where, for every $[c_i, c_j] \in \mathbb{I}(\mathbb{C}_B)^+$ and $p \in \mathcal{AP}$, $p \in V([c_i, c_j])$ iff $(p, [c_i, c_j]) \in \Phi(B)$. We show by induction on ψ that, for every $(\psi, [c_i, c_j]) \in S(B)$, $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$.

We reason by induction on the complexity of ψ:

- Let $\psi = \pi$ (resp., $\psi = \neg\pi$). Since $(\pi, [c_i, c_j]) \in S(B)$ (resp., $(\neg\pi, [c_i, c_j]) \in S(B)$) and B is open, then $c_i \neq c_j$ (resp., $c_i = c_j$). Hence $\mathbf{M}^+, [c_i, c_j] \Vdash \pi$ (resp., $\mathbf{M}^+, [c_i, c_j] \Vdash \neg\pi$);
- Let $\psi = p$ or $\psi = \neg p$ where $p \in \mathcal{AP}$. Then the claim follows by definition, because if $(\neg p, [c_i, c_j]) \in S(B)$ then $(p, [c_i, c_j]) \notin S(B)$ since B is open;
- Let $\psi = \neg\neg\xi$. Then by Corollary 3, $(\xi, [c_i, c_j]) \in S(B)$, and by inductive hypothesis $\mathbf{M}^+, [c_i, c_j] \Vdash \xi$. So $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Let $\psi = \xi_0 \wedge \xi_1$. Then by Corollary 3, $(\xi_0, [c_i, c_j]) \in S(B)$ and $(\xi_1, [c_i, c_j]) \in S(B)$. By inductive hypothesis, $\mathbf{M}^+, [c_i, c_j] \Vdash \xi_0$ and $\mathbf{M}^+, [c_i, c_j] \Vdash \xi_1$, so $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Let $\psi = \neg(\xi_0 \wedge \xi_1)$. Then by Corollary 3, $(\neg\xi_0, [c_i, c_j]) \in S(B)$ or $(\neg\xi_1, [c_i, c_j]) \in S(B)$. By inductive hypothesis $\mathbf{M}^+, [c_i, c_j] \Vdash \neg\xi_0$ or $\mathbf{M}^+, [c_i, c_j] \Vdash \neg\xi_1$, so $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Let $\psi = \xi_0 C \xi_1$. Then by Corollary 3, $(\xi_0, [c_i, c]) \in S(B)$ and $(\xi_1, [c, c_i]) \in S(B)$ for some $c \in \mathbb{C}_B$ such that $c_i \leqslant c \leqslant c_j$. Thus, by inductive hypothesis, $\mathbf{M}^+, [c_i, c] \Vdash \xi_0$ and $\mathbf{M}^+, [c, c_j] \Vdash \xi_1$, and thus $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Similarly for $\psi = \xi_0 D \xi_1$ and $\psi = \xi_0 T \xi_1$;
- Let $\psi = \neg(\xi_0 C \xi_1)$. Then by Corollary 3, for all $c \in \mathbb{C}_B$ such that $c_i \leqslant c \leqslant c_j$, $(\neg\xi_0, [c_i, c]) \in S(B)$ and $(\neg\xi_1, [c, c_j]) \in S(B)$. Hence, by the inductive hypothesis, $\mathbf{M}^+, [c_i, c] \Vdash \neg\xi_0$ and $\mathbf{M}^+, [c, c_j] \Vdash \neg\xi_1$, for all $c_i \leqslant c \leqslant c_j$. Thus, $\mathbf{M}^+, [c_i, c_j] \Vdash \psi$;
- Similarly for $\psi = \neg(\xi_0 D \xi_1)$ and $\psi = \neg(\xi_0 T \xi_1)$.

This completes the induction. In particular, we obtain that $\neg\phi$ is satisfied in $\mathbf{M}^+$, which is in contradiction with the assumption that ϕ is valid. □

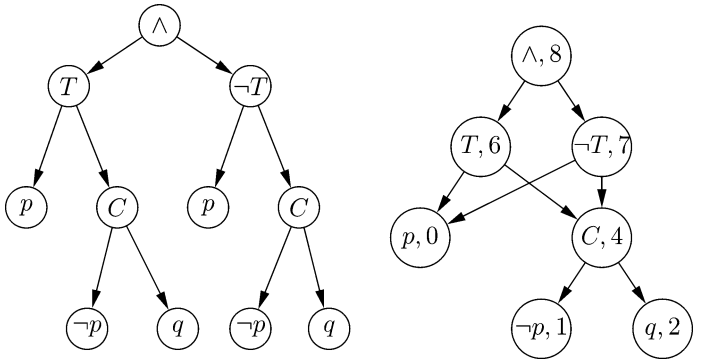


Fig. 5. The tree for the formula $\phi = (pT(\neg pCq)) \wedge \neg(pT(\neg pCq))$.

5. An implementation of the tableau method

In this section, we describe the main features of our implementation of the proposed tableau method, which has been successfully used for testing a number of BCDT$^+$-formulas [28].

Input formula. The input BCDT$^+$-formula is arranged in a tree structure whose internal nodes contain a Boolean connective or a temporal operator and whose leaves contain a literal. As an example, the formula $\phi = (pT(\neg pCq)) \wedge \neg(pT(\neg pCq))$ generates the tree depicted in Fig. 5, left. The subtrees of the tree for a formula ϕ identify the subformulas of ϕ. Obviously, identical subformulas give raise to identical subtrees. To obtain a more compact representation, we collapse identical subtrees, thus turning the tree into a direct acyclic graph (DAG). To this end, we define a recursive procedure that enumerates the distinct subformulas and generates a DAG whose nodes contain a subformula labeled by a natural number. We constrain such a labeling procedure to associate even (resp. odd) numbers with positive (resp. negative) subformulas; moreover, if the positive formula ψ is labeled by $n - 1$, the corresponding negative subformula $\neg\psi$ (if present) is labeled by n. This allows us to check whether two subformulas are identical, or whether one of them is the negation of the other, by comparing two natural numbers. The application of this enumeration procedure to the above formula ϕ produces the DAG of Fig. 5, right. Furthermore, if a formula ϕ has h distinct subformulas, we can use an array of at most $2 \cdot h$ bits to compactly represent any subset of its subformulas: we set the nth bit of the array to 1 if and only if the subformula labeled by $n - 1$ belongs to the subset.

Branch management. Any single branch of the tableau for an input formula ϕ is managed according to a depth-first strategy. A branch is (recursively) expanded until it is closed (in such a case, the procedure backtracks and it starts again with the next branch, if any) or it is open, but saturated (in such a case, the method returns a model for the input formula). Obviously it may happen that the expansion goes forever with the branch neither closed nor (open and) saturated. A single branch contains: (i) a list L of pairs $(\psi, [c_i, c_j])$, where ψ is a subformula of ϕ and $[c_i, c_j]$ is an interval; (ii) a (possibly empty) sublist L' of L that

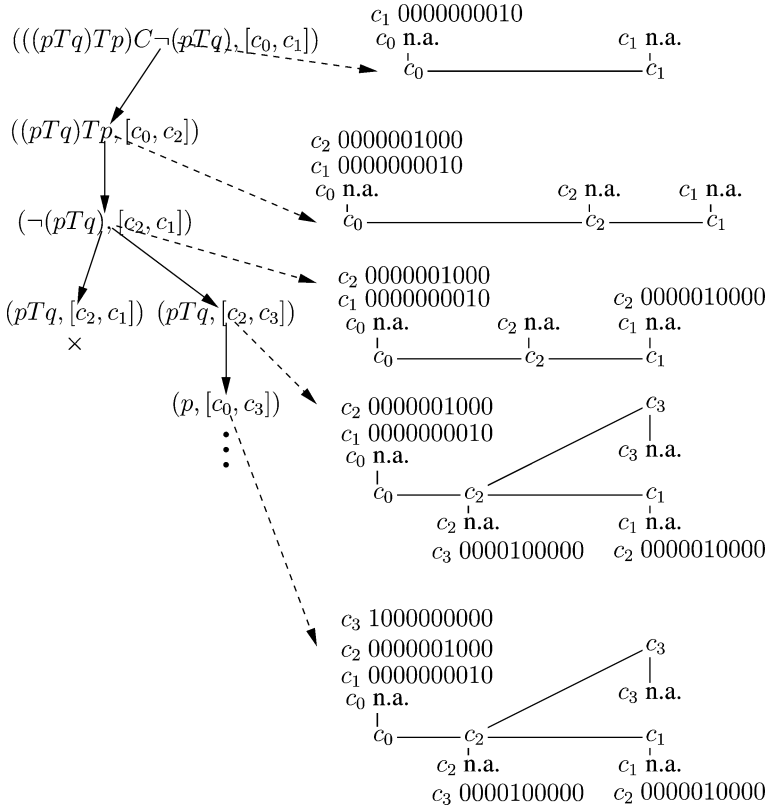


Fig. 6. An example of interval structure.

contains only those pairs $(\psi, [c_i, c_j])$, where ψ is a formula of the form $\neg C$, $\neg D$, or $\neg T$ (universal formula); (iii) a graph G that represents the current interval structure associated with (the leaf of) the branch. Furthermore, a current position is defined for the list L. All elements of L to the left of the current position identify the expansion steps applied to the branch so far, while those to the right (including that in the current position) correspond to expansion steps that have never been executed yet. L' includes the universal subformulas of ϕ that can possibly be reused during some subsequent expansion step that extends the interval structure. The nodes of G correspond to the endpoints of the intervals belonging to the current interval structure (such a structure must be updated whenever an expansion step introduces a new endpoint). Every node of G is labeled by a unique identifier c_i and provided with two lists of pointers to its immediate successors and predecessors (cf. Fig. 6).

The problem of keeping track of the subformulas of ϕ associated with (true over) the various intervals of the current interval structure is dealt with as follows. For every node (labeled by) c_i, let c_j, with $i \leqslant j$, be the label with maximum index for which there exists a subformula which is true over $[c_i, c_j]$, or over $[c_j, c_i]$ (if any). We associate an array $[c_i, c_{i+1}, \ldots, c_{j-1}, c_j]$ with the node c_i. Moreover, for every c_k belonging to this array, if there exists a subformula which is true over $[c_i, c_k]$ (or $[c_k, c_i]$), we provide c_k with an array

of $2 \cdot h$ bits, where h is the number of distinct subformulas of ϕ. For every $1 \leqslant n \leqslant 2 \cdot h$, the array has value 1 in the nth position if (and only if) the subformula of ϕ labeled by $n-1$ is true over $[c_i, c_k]$ (or $[c_k, c_i]$). Such an encoding allows one to keep track of the truth of a subformula over a given interval and to possibly detect a contradiction, as well as to withdraw the truth of a subformula over an interval during backtracking, in constant time. Finally, to efficiently deal with the labeled universal subformulas $(\psi, [c_i, c_j])$ in L', we maintain an array of successors of c_j (resp. predecessors of c_i, elements between c_i and c_j), that can be easily updated whenever a new endpoint is added to the interval structure. Taking advantage of these arrays, one can easily determine the universal subformulas associated with the branch which are activated by the addition of the endpoint.

To illustrate the management of the interval structure, in Fig. 6 we describe the effects of the application of some expansion steps to a branch of a tableau for the formula $\phi = ((pTq)Tp)C\neg(pTq)$. We assume the subformulas of ϕ to be labeled as follows: $label(p) = 0$, $label(\neg p) = 1$, $label(q) = 2$, $label(\neg q) = 3$, $label(pTq) = 4$, $label(\neg(pTq)) = 5$, $label((pTq)Tp) = 6$, $label(\neg((pTq)Tp)) = 7$, $label(\phi) = 8$, and $label(\neg\phi) = 9$. At the first step, the interval structure consists of two nodes (labeled by) c_0 and c_1, with $c_0 < c_1$. The arrays $[c_0, c_1]$ and $[c_1]$ are associated with c_0 and c_1, respectively. Since $0 < 1$, to constrain the formula ϕ, with $label(\phi) = 8$, to hold over $[c_0, c_1]$, we provide the entry c_1 of the c_0-labeled node with a 10-bits array whose 9th bit is set to 1. Since there are not formulas associated with the intervals $[c_0, c_0]$ and $[c_1, c_1]$, there are not 10-bits arrays corresponding to the entry c_0 for the c_0-labeled node and to the entry c_1 for the c_1-labeled node. In Fig. 6, we denote this situation by the expression n.a., which stands for not allocated. At the second step, we extend the branch by applying the expansion rule for C to the formula ϕ. Accordingly, we add a node c_2, with $c_0 < c_2 < c_1$, to the current interval structure. Since $(pTq)Tp$ (resp. $\neg(pTq)$) holds over $[c_0, c_2]$ (resp. $[c_2, c_1]$), and $0 < 2$ (resp., $1 < 2$), we provide the entry c_2 of the c_0-labeled (resp. c_1-labeled) node with a 10-bits array whose 7th (resp. 6th) bit is set to 1. The next step shows how a branch can be closed. Since c_1 is a successor of c_2, the application of the expansion rule for T to the formula $(pTq)Tp$ may result in the request for pTq (resp. p) to hold over $[c_2, c_1]$ (resp. $[c_0, c_1]$). Since $1 < 2$, the 10-bits array associated with the entry c_2 of the c_1-labeled node must be updated by setting its 5th bit to 1. This means that we require both pTq and its negation to hold over $[c_2, c_1]$ (contradiction). Such a contradiction can be immediately detected: in the 10-bits array associated with the entry c_2 of the c_1-labeled node there exist two consecutive bits, the first one being in an odd position (in the example, the 5th and the 6th one), both set to 1. Once the contradiction has been detected, the procedure backtracks, and it explores alternative expansions of the branch (if any). In the example, it chooses to add a new node c_3, with $c_2 < c_3$, incomparable with c_1.

Experimental results. We developed a *C*-implementation of the proposed tableau method for BCDT$^+$, and we carried out some tests on an Intel x86 machine, with a 2GHz Pentium 4 CPU, 40 Gb Hard Drive serial-ATA 150, and 1Gb of DDR SDRAM. We also compared its performances with those of the well-known first-order theorem prover Spass [15] (installed on the same machine), which takes advantage of a special form of syntactic unification. To make the comparison possible, BCDT$^+$ formulas have been mapped into their first-order counterparts (in a language with a binary relation symbol $<$ which has been constrained to

Table 1

BCDT$^+$-formula	Our implementation	Spass
$\neg(\neg(\phi T \psi) C \phi \rightarrow \neg\psi)$	< 1	30
$\neg(\neg(\phi T \psi) D \psi \rightarrow \neg\phi)$	10	41
$\neg(\pi C \phi \leftrightarrow \phi)$	9	29
$\neg(\pi T \phi \leftrightarrow \phi)$	11	32
$((pTq)Tp)C\neg(pTq)$	10	26
$(p \rightarrow (pT\pi))$	10	23

be a partial ordering). A comparison of the performances of the two systems on some meaningful unsatisfiable/satisfiable BCDT$^+$ formulas (execution time is measured in msec) is given in Table 1.

6. Conclusions

In this paper, we described a general tableau method for CDT logic, interpreted over partial orders, which combines features of the classical tableau method for first-order logic with those of explicit tableau methods for modal logics with constraint label management. The method can be easily tailored to most existing propositional interval temporal logics. We proved its soundness and completeness, and we provided it with an efficient implementation in *C*. We are currently looking for meaningful syntactic (fragments of the logic) and/or semantic (classes of interval structures) fragments where the tableau terminates, thus providing a decision procedure.

Acknowledgements

The authors would like to thank the Italian Ministero degli Affari Esteri and the National Research Foundation of South Africa for the research grant, under the Joint Italy/South Africa Science and Technology Agreement, that they received for the project: "Theory and applications of temporal logics to computer science and artificial intelligence".

References

[1] P. Abate, R. Goré, System description: The tableaux workbench, in: Proc. of the International Conference TABLEAUX 2003, in: Lecture Notes in Artif. Intell., vol. 2796, Springer, Berlin, 2003, pp. 230–236.
[2] J. Allen, Maintaining knowledge about temporal intervals, Comm. ACM 26 (11) (1983) 832–843.
[3] B. Beckert, R. Goré, Free-variable tableaux for propositional modal logics, Studia Logica 69 (1) (2001) 59–96.
[4] H. Bowman, S. Thompson, A decision procedure and complete axiomatization of finite interval temporal logic with projection, J. Logic Comput. 13 (2) (2003) 195–239.
[5] S. Cerrito, M. Cialdea-Mayer, Bounded model search in linear temporal logic and its application to planning, in: Proc. of the International Conference TABLEAUX 1998, in: Lecture Notes in Artif. Intell., vol. 1397, Springer, Berlin, 1998, pp. 124–140.

[6] S. Cerrito, M. Cialdea-Mayer, S. Praud, First-order linear temporal logic over finite time structures, in: H. Ganzinger, D. McAllester, A. Voronkov (Eds.), Proc. of the 6th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, in: Lecture Notes in Artif. Intell., vol. 1705, Springer, Berlin, 1999, pp. 62–76.
[7] N. Chetcuti-Serandio, L.F. del Cerro, A mixed decision method for duration calculus, J. Logic Comput. 10 (2000) 877–895.
[8] M. D'Agostino, D. Gabbay, R. Hähnle, J. Posegga (Eds.), Handbook of Tableau Methods, Kluwer Academic Press, 1999.
[9] E. Emerson, J. Halpern, Decision procedures and expressiveness in the temporal logic of branching time, J. Comput. System Sci. 30 (1) (1985) 1–24.
[10] M. Fitting, Proof Methods for Modal and Intuitionistic Logics, vol. 169, D. Reidel, Dordrecht, 1983.
[11] V. Goranko, A. Montanari, G. Sciavicco, Propositional interval neighborhood temporal logics, J. Universal Comput. Sci. 9 (9) (2003) 1137–1167.
[12] V. Goranko, A. Montanari, G. Sciavicco, A general tableau method for propositional interval temporal logics, in: Proc. of the International Conference TABLEAUX 2003, in: Lecture Notes in Artif. Intell., vol. 2796, Springer, Berlin, 2003, pp. 102–116.
[13] V. Goranko, A. Montanari, G. Sciavicco, A road map of interval temporal logics and duration calculi, J. Appl. Non-Class. Logics 14 (1–2) (2004) 9–54.
[14] E. Grädel, C. Hirsch, M. Otto, Back and forth between guarded and modal logics, ACM Trans. Comput. Logics 3 (3) (2002) 418–463.
[15] U. Hustadt, R.A. Schmidt, MSPASS: Modal reasoning by translation and first-order resolution, in: Proc. of the International Conference TABLEAUX 2000, in: Lecture Notes in Artif. Intell., vol. 1847, Springer, Berlin, 2000, pp. 67–71.
[16] R. Hähnle, O. Ibens, Improving temporal logic tableaux using integer constraints, in: Proc. of the 1st International Conference on Temporal Logic, in: Lecture Notes in Comput. Sci., vol. 827, Springer, Berlin, 1994, pp. 535–539.
[17] J. Halpern, Y. Shoham, A propositional modal logic of time intervals, J. ACM 38 (4) (1991) 935–962.
[18] Y. Kesten, Z. Manna, H. McGuire, A. Pnueli, A decision algorithm for full propositional temporal logic, in: Proc. of the 5th International Conference on Computer Aided Verification, 1993, pp. 97–109.
[19] S. Kono, A combination of clausal and non-clausal temporal logic programs, in: M. Fisher, R. Owens (Eds.), Executable Modal and Temporal Logics, in: Lecture Notes in Comput. Sci., vol. 897, Springer, Berlin, 1995, pp. 40–57.
[20] R. Kontchakov, C. Lutz, F. Wolter, M. Zakharyaschev, Temporalizing tableaux, Studia Logica 76 (1) (2004) 91–134.
[21] M. Marx, Y. Venema, Multi-Dimensional Modal Logics, Kluwer Academic Press, Dordrecht, 1997.
[22] B. Moszkowski, Reasoning about digital circuits, PhD thesis, Department of Computer Science, Stanford University, Technical Report STAN-CS-83-970, Stanford, CA, 1983.
[23] L. Nguyen, Analytic tableau systems for propositional bimodal logics of knowledge and belief, in: Proc. of the International Conference TABLEAUX 2002, in: Lecture Notes in Artif. Intell., vol. 2381, Springer, Berlin, 2002, pp. 206–220.
[24] L. Paulson, A generic tableau prover and its integration with Isabelle, J. Universal Comput. Sci. 5 (3) (1999) 73–87.
[25] A. Pnueli, R. Sherman, Semantic tableau for temporal logic, Technical Report CS81-82, The Weizmann Institute, 1981.
[26] O. Lichtenstein, A. Pnueli, Propositional temporal logic: Decidability and completeness, Logic J. IGPL 8 (1) (2000) 55–85.
[27] W. Rautenberg, Modal tableau calculi and interpolation, J. Philos. Logics 12 (1983) 403–423.
[28] P. Sala, Tableau systems for interval temporal logics, Tesi di Laurea in Informatica, Università di Udine, 2003 (in Italian).
[29] P. Schmitt, J. Goubault-Larrecq, A tableau system for linear-time temporal logic, in: E. Brinksma (Ed.), Proc. of the 3rd Workshop on Tools and Algorithms for the Construction and Analysis of Systems, in: Lecture Notes in Comput. Sci., vol. 1217, Springer, Berlin, 1997, pp. 130–144.
[30] P. Schmitt, J. Goubault-Larrecq, A tableau system for full linear temporal logic, Draft, available at: http://www.dyade.fr/fr/actions/vip/jgl/ltl2.ps.gz, Technical Report, 1997.

[31] A. Sistla, E. Clarke, The complexity of propositional linear time temporal logics, J. ACM 32 (3) (1985) 733–749.
[32] Y. Venema, A modal logic for chopping intervals, J. Logic Comput. 1 (4) (1991) 453–476.
[33] P. Wolper, The tableau method for temporal logic: An overview, Logique et Analyse 28 (1985) 119–136.
[34] M. Wooldridge, C. Dixon, M. Fisher, A tableau-based proof method for temporal logics of knowledge and belief, J. Appl. Non-Class. Logics 8 (3) (1998) 225–258.
[35] C. Zhou, C. Hoare, A.P. Ravn, A calculus of durations, Inform. Process. Lett. 40 (5) (1991) 269–276.
[36] C. Zhou, M.R. Hansen, An adequate first order interval logic, in: W. de Roever, H. Langmaak, A. Pnueli (Eds.), Compositionality: The Significant Difference, in: Lecture Notes in Comput. Sci., vol. 1536, Springer, Berlin, 2003, pp. 584–608.